Dynamic HDR Environment Capture for Mixed Reality

David R. Walton
Department of Computer Science
University College London
david.walton.13@ucl.ac.uk

Anthony Steed
Department of Computer Science
University College London
a.steed@ucl.ac.uk

Figure 1: Examples of MR frames with a variety of virtual objects rendered using the proposed approach

ABSTRACT

Rendering accurate and convincing virtual content into mixed reality (MR) scenes requires detailed illumination information about the real environment. In existing MR systems, this information is often captured using light probes [1, 8, 9, 17, 19–21], or by reconstructing the real environment as a preprocess [31, 38, 54]. We present a method for capturing and updating a HDR radiance map of the real environment and tracking camera motion in real time using a self-contained camera system, without prior knowledge about the real scene. The method is capable of producing plausible results immediately and improving in quality as more of the scene is reconstructed. We demonstrate how this can be used to render convincing virtual objects whose illumination changes dynamically to reflect the changing real environment around them.

CCS CONCEPTS

• **Computing methodologies** → **Computational photography**; **Mixed / augmented reality**; **Point-based models**; *Reflectance modeling*;

KEYWORDS

Mixed Reality, 3D Reconstruction, HDR

ACM Reference Format:
David R. Walton and Anthony Steed. 2018. Dynamic HDR Environment Capture for Mixed Reality. In *VRST 2018: 24th ACM Symposium on Virtual Reality Software and Technology (VRST '18), November 28-December 1, 2018, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3281505.3281531

1 INTRODUCTION

Techniques such as differential rendering [8] have enabled the rendering of extremely realistic virtual content into mixed reality (MR) scenes, provided the real world lighting, geometry and material information is known. In practice, acquiring this information is often a significant challenge and restricts the situations in which realistic MR can be implemented. Many works in the field focus on addressing this challenge (see section 2).

In this paper, we present a novel approach for capturing real-world lighting and geometry that combines a number of features desirable for MR. Working in unprepared environments, it produces radiance estimates immediately and improves them over time. The lighting and geometry is updated dynamically as more information becomes available, reflecting lighting changes in the real environment. The approach uses a system of cameras contained in a single device and does not require external light probes. No particular restrictions are placed on the geometry of the real scene. Finally, the approach tracks the motion of the camera system in the real world, which can then be used to render virtual content into the camera output for video see-through MR.

The method presented here uses a camera system comprising an RGBD camera, together with one or more colour cameras mounted in a single unit, such as a mobile device. The output of the cameras is used to generate a detailed 3D reconstruction of the real world via a dense SLAM approach, and this reconstruction is updated in real time using the output of all cameras in the system. Estimated distant lighting is captured in environment map keyframes, enabling a complete spherical light probe to be reconstructed even when the geometric scene model is incomplete.

Our method builds upon the surfel-based dense SLAM approach proposed by Keller et al. [24] and adds the following contributions:

(1) A method for updating the surfel-based model using the output of a multi-camera system, capturing HDR colours and updating them in real time.
(2) The generation, completion, updating and rendering of environment map keyframes, which store estimates of the distant real environment.
(3) A prototype video-see through MR application which applies our approach, using a two-camera system to capture the real scene and render realistic virtual content.

VRST '18, November 28-December 1, 2018, Tokyo, Japan
2018. ACM ISBN 978-1-4503-6086-9/18/11…$15.00
https://doi.org/10.1145/3281505.3281531

2 RELATED WORK

An important goal of MR is to render virtual content that is not only realistic, but also appears seamlessly integrated with the real scene. This requires detailed geometric information, accurate camera tracking and lighting information about the real scene. If the MR approach is to work in unprepared environments, these data must be captured and updated in real time.

2.1 Dense RGBD SLAM

Beginning with the publication of Kinect Fusion [33] and the availability of affordable RGBD sensors such as the Kinect, there has been a rapid increase in the development of GPU-accelerated dense 3D reconstruction methods using these devices. These methods track the motion of the RGBD camera in the world and integrate depth data into a dense 3D geometric model. The majority of recent approaches can roughly be grouped according to the type of world representation they use, into volumetric or surfel-based (sometimes called point-based) approaches.

Volumetric approaches include the original Kinect Fusion approach. These approaches store the world representation as samples of an implicit function, typically a variety of signed distance function, on a regular grid. Kinect Fusion used a single fixed grid which only allowed reconstruction within a fixed volume, and was fairly memory inefficient as much of the volume typically consists of empty space. More recent publications have addressed both of these shortcomings. Moving-volume approaches [39, 48] extend the area which can be reconstructed by shifting the reconstructed volume as the sensor moves in the real world, and serialising already captured areas. More recent approaches such as [18, 34, 53] instead address the problem by storing the volume more efficiently, using volume hierarchies or hashing. Dai et al. [6] recently proposed an approach which allows the quality of previously observed regions to be refined by de-integrating and re-integrating data. All volumetric approaches, however, require significant computation to integrate the input data into the volume, and to render the volume for tracking and visualisation.

Surfel-based approaches such as that of Keller et al. [24] instead use a model consisting of a list of points with associated normals and radii, each defining a small disc which forms part of a real surface. Surfels have long been used as an alternative to connected triangle meshes for representing 3D objects in computer graphics [36]. As compared to volumetric approaches, surfel-based approaches are inherently more memory-efficient as all data stored corresponds to part of a real surface. Creating, updating and rendering surfels is also comparatively computationally inexpensive, as no expensive raycasting operations are required. As compared to volumetric approaches, however, converting the resulting model into a triangle mesh is more complex, as a marching cubes approach cannot be used directly. More recent surfel-based dense SLAM approaches have added capabilities such as real-time loop closure [49] and the ability to explicitly detect and handle planar surfaces [40].

2.2 Real-world Lighting Capture

In addition to geometric information and camera tracking, real world illumination information is also required in order to render the virtual content consistently with the real scene.

Many previous approaches make use of light probes to capture incident illumination at the virtual object location. This can then be used as input to a differential rendering algorithm to produce the virtual content. Probes can be roughly categorised as passive or active.

Passive probes are objects with known reflectance properties. Images of these are captured using a separate camera, and from these images the surrounding lighting can be inferred. Chrome spheres are commonly used [1, 8, 9, 17, 21], but diffuse and non-spherical probes can also be used [2, 4, 51]. Recently, Meka et al. [32] presented an approach for material estimation of real objects in monocular video which also produces an environment lighting estimate. This can in effect allow objects with unknown material properties to be used as light probes.

Active probes are generally imaging devices. Cameras with wide fields of view (e.g. fisheye lenses [19, 20]) or arrays of multiple cameras are often employed to capture more of the surrounding lighting.

Placing a light probe at the virtual object location can be impractical or impossible in some situations, so other approaches focus on removing this requirement. Some approaches process information from light probes placed elsewhere in the scene to infer lighting at the virtual object location [11, 37].

Other approaches remove the requirement for light probes entirely, and recover the lighting indirectly in other ways. Many such approaches fit a lighting model to the observed real world using inverse rendering style techniques. Approaches include identifying shadows in the real image, and using these to infer light source directions [3, 16, 41–43, 55] or spherical harmonic (SH) lighting coefficients [35]. Karsch et al. [22, 23] instead use a machine learning approach to estimate rough geometry from the input image, and select a plausible lighting environment from a database of real examples.

Other recent approaches use the output of an RGBD camera, and can take advantage of the additional depth information to better infer the real lighting. Gruber et al. [13–15] reconstruct part of the real scene near the virtual content using a dense SLAM approach, and then use this to infer SH lighting coefficients. Whelan et al. [50] track the motion of specular highlights whilst reconstructing the environment using a dense SLAM approach, and use this information to infer the location of point light sources in the environment.

Finally, some approaches such as that of Meilland et al. [31] use an adapted dense RGBD SLAM approach to capture a full HDR model of the environment. This then provides all the geometric and lighting information required to insert the virtual content. Zhang et al. [54] showed how this kind of approach could be used in an MR application for virtually refurnishing rooms, and Rohmer et al. [38] used a similar HDR model to render photorealistic virtual content on mobile and desktop hardware. These approaches capture rich scene information using just an RGBD sensor, but have the disadvantage of requiring the whole scene to be reconstructed before virtual content can be added. They also are limited in their ability to update the model should the real lighting or geometry change. Finally, they can only be used in settings where the environment can be reconstructed using an RGBD sensor (i.e. small indoor environments).

The approach most similar to the one presented here is [47], which uses a similar hardware setup, effectively combining an RGBD sensor with an upward-facing light probe (fisheye camera). This enables a full scene lighting model to be reconstructed instantly and updated to reflect changes in the environment. This previous approach had a number of shortcomings which are addressed by the approach presented here. Most notably, [47] required rough scene geometry and an initial camera pose a priori, wheras the presented approach requires no prior information about the scene. There were also restrictions on the shape of the rough geometry which meant it could only be used in certain indoor environments. Furthermore, the lighting captured was also low dynamic range, which limited the accuracy of the results, leading to artefacts such as soft shadows and inaccurate specular highlights; all of these issues are addressed by the presented approach.

3 APPROACH OVERVIEW

The presented approach reconstructs a model consisting of two parts. The first is a dense surfel-based reconstruction which captures detailed geometry and appearance near the virtual objects. The second consists of a number of environment map keyframes, each of which consists of a HDR cubemap and associated 3D location. These capture the distant lighting.

The surfels used in our model are similar to those used in other approaches, such as that of Keller et al. [24]. Each consists of a 3D point, normal and radius, which define a small disc in 3D space. Some additional information is also stored for each surfel: a colour, and a confidence value, related to how many times it has been observed and updated. Our surfels differ from those in Keller et al.'s approach in that they store high dynamic range (HDR) colour values using floating-point datatypes. The surfels are constructed using images from the RGBD camera, and subsequently updated using data from both cameras. Camera tracking is achieved by aligning new RGBD frames to the existing model.

The environment map keyframes each consist of a cubemap, again consisting of HDR colour values, together with an associated 3D location. The alpha channel of the environment map is set aside to store additional per-pixel information (further details are given in section 4.4). These keyframes are created and updated using images captured by the camera system, and completed with plausible values using an inpainting approach. New keyframes are created as necessary as the device moves in the real scene. When new surfels are created, the most recent keyframe can also be used to provide accurate HDR colours when this is appropriate.

Figure 2 contains a flowchart giving an overview of the approach, including images of inputs and outputs, and a visualisation of the reconstruction.

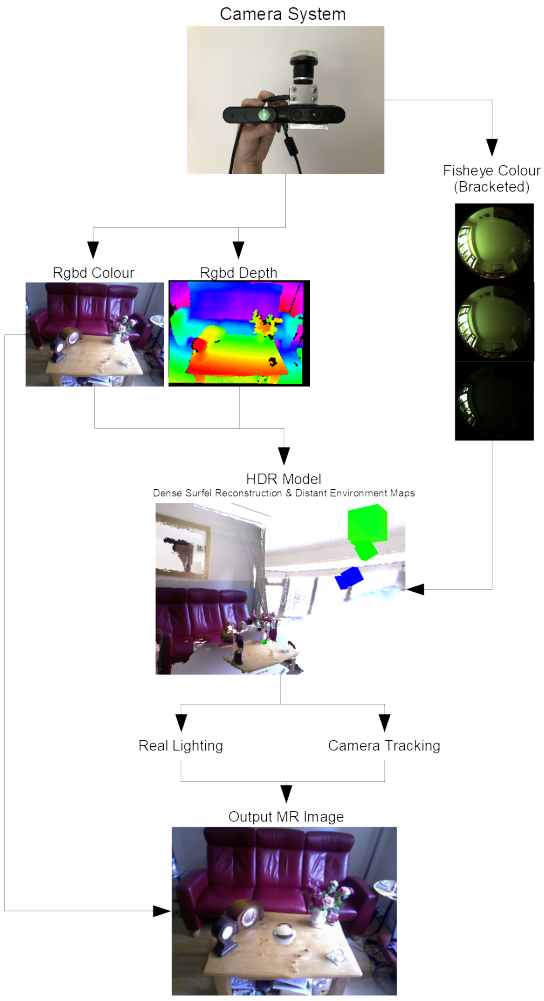

Figure 2: Flowchart giving an overview of the proposed approach

4 APPROACH DETAIL

This section describes each component of the presented approach in further detail.

4.1 Hardware

The approach detailed here is intended to be applicable to a variety of possible camera systems, containing an RGBD camera of some description and one or more colour cameras. Our prototype hardware consists of an Asus Xtion Pro RGBD camera, joined to an upward-facing Point Grey CM3-U3-31S4C-CS camera, fitted with a Lensagon CF5M1414 fisheye lens. These are connected to a desktop PC which performs processing and rendering, and displays the results. In future implementations we envision the cameras being mounted in a self-contained device such as a mobile phone or MR head-mounted display (HMD).

The camera system needed to be calibrated. The intrinsic calibration of the fisheye and RGBD cameras was required, in addition to the extrinsic rigid transform between them. The colour camera in the RGBD sensor was calibrated using a standard calibration toolbox [30]. The camera had a depth-to-colour registration feature, which was used to ensure the depth images could be processed using the same calibration. The fisheye camera was calibrated using

Scaramuzza et al.'s toolbox [44]. This used a traditional calibration approach based on capturing several images of a checkerboard pattern. At runtime the fisheye images were first undistorted to fit a simpler omnidirectional camera model by Ying and Hu [52]. This was done on the GPU using a fast lookup table (LUT) approach. The parameters for this model were selected manually to provide a similar field of view and image resolution to the true camera model. The simplified camera model made the subsequent stages in the approach which used the fisheye images faster. This two-model approach was similar to that used by Caruso et al. [5] to simplify input to fisheye-based SLAM. Finally, the transform between the cameras was found by capturing an image with both cameras, manually selecting point matches and solving for the transform best aligning the matches using an optimisation approach [27, 28].

We use the colour output of the RGBD camera as the basis for the video see-through MR output. Since it will be directly observed by the user, visual quality is important. Because of this, the auto exposure and auto white-balance features of the Xtion Pro camera were enabled.

The other cameras in the system are mainly used to aid in obtaining HDR colours for the scene, and the images they produce are not directly presented to the end-user. In our case, we made use of bracketing to capture the full dynamic range of the scene. The fisheye camera used in the prototype had a HDR feature, which cycled through four different exposure settings whilst capturing frames at video framerate. The framerate of the camera was locked to mains frequency (50Hz/60Hz, depending upon location) in order to avoid issues with flickering of light fixtures in real scenes.

The four exposure settings of the fisheye camera needed to be such that the full dynamic range of the real scene could be captured. We found that typically, 3 exposure settings were sufficient to capture the dynamic range of the scenes we tested. As such, we cycled through 3 settings: Low, Medium and High exposure, in LMHM order (i.e. with two medium exposures in each 4-frame group). A simplified auto exposure method was implemented to select these exposures. This fixed the medium exposure to a predetermined value, and modified the low and high exposures to avoid saturated/black pixels, whilst maintaining an overlap in the dynamic ranges of the three settings.

After each fisheye camera image is captured, it is undistorted as detailed above. Following this, it is converted from LDR to HDR. The fisheye camera is set to produce linear output, and white balance is disabled. Consequently, this just involves removing the effects of exposure time and gain by multiplying by a scalar $s = \frac{1}{eg}$, where e is exposure time in seconds and g is the gain expressed as a linear scaling factor (our camera specified gain in decibels, so g was calculated as $g = 10^{\frac{d}{20}}$).

Unfortunately, the Xtion Pro camera API did not allow querying the exposure settings necessary to convert the colour output it provided to linear, HDR values. We therefore estimated these settings by comparing pixels observed by both the fisheye camera and RGBD camera. Since the fisheye camera's exposure settings were known, this allowed us to infer the settings for the RGBD camera. This process is detailed in the following section.

4.2 HDR Surfel-based Reconstruction

As mentioned in section 3, we used a surfel-based dense SLAM approach to generate our model, similar to that proposed by Keller et al. [24], but adapted to capture HDR colours. The parts common to both approaches will briefly be outlined, and then the specific modifications made to capture HDR colour values will be detailed.

The model of the environment consists of a list of surfels, each consisting of a point, normal and radius, along with a confidence counter and HDR colour value. The surfel list is initially empty. When a new frame is captured by the RGBD camera, it is projected to a number of surfels, each corresponding to a pixel in the depth map. At the first frame, these new surfels are simply added to the list.

On subsequent frames, the pose of the RGBD camera relative to the model is first determined using the camera tracking approach detailed in section 4.5. Following the tracking stage, the frame is again projected to a new set of surfels. Each new surfel can then either be added to the list, or used to update an existing surfel.

If a new surfel corresponds to an existing surfel (i.e. is close in 3D space, has a similar normal and radius) it is used to update the existing surfel. The update consists of taking a weighted average of the surfel's properties based on the existing surfel's confidence. This confidence value is incremented on each successful update. If the new surfel does not correspond to any existing surfels, it is added to the list of surfels.

Due to sensor noise, for example, "bad" surfels may be added via this process that do not correspond to geometry in the real scene. For this reason, surfels below a confidence threshold are considered unstable, and not used for tracking. If a surfel remains unstable for a period of time, it is deleted from the list.

In order to quickly determine which new surfels correspond to existing ones, an index map is used. This is an image containing the existing surfels, rendered from the camera's viewpoint. Each is rendered into a 32-bit integer texture as a single pixel-width point, with the colour set to be the index of the surfel in the main surfel list. This map is consulted when updating the existing surfels, to quickly find possible correspondences. The index map is rendered at 4x the depth map resolution on each axis (i.e. 16x the number of pixels) and each new surfel checks the corresponding 4x4 window for up to 16 possible existing surfels to find matches.

Before projecting a new set of surfels corresponding to the current frame, we first convert the observed LDR colour values to HDR values in a linear colour space. We approximately linearise the input values by applying an inverse gamma correction ($\gamma = 2.2$), similarly to [38, 54]. We then estimate the exposure and white balance applied by the RGBD colour camera. As in [38], we model the exposure and white balance applied by the camera as a linear scaling applied to each colour channel.

Specifically, we start from a sparse set of samples in the upper region of the RGBD image, which are in the region also observed by the fisheye camera. For each of these samples, we do the following:

- If a valid depth value is available here and the colour is not saturated, reverse-project to the corresponding 3D point.
- Project this 3D point into the fisheye image. If it projects to a valid image location, sample the fisheye image.

- If the fisheye sample is also not saturated or black, add the fisheye and RGBD colour samples to a list.

This gives a set of valid sample pairs, each of which has a sample of the linearised RGBD image l_i and a corresponding sample of the HDR fisheye image f_i. The exposure and white balance scaling e can then be estimated by calculating $e \approx \frac{\sum f_i}{\sum l_i}$. In the event that a sufficient number of valid sample pairs cannot be found (due to missing depth values or saturated/black pixels) the current estimate of e is not updated, as the new estimate is likely to be unreliable.

This extra step did not take a significant proportion of the computation time, as only a small proportion of the total needed to be compared each frame to obtain a reasonably accurate estimate. If this approach were implemented using an RGBD camera which provided auto exposure and white balance information, this step would not be necessary, but many cameras on RGBD sensors and mobile devices are currently limited in this way.

Once e is determined the image can then be converted to HDR. When converting, we also store a flag for each input pixel, indicating if it was saturated or below the blackpoint in the input LDR image. This information is used to assign a lower initial confidence to these surfels, and to avoid them during the colour camera tracking step.

4.3 Updating Surfel Colours

Our approach also includes a method for other cameras in the system to update the colours of surfels in the dense model of the environment. This allows their true intensity to be determined, in the event that they were saturated/black in the RGBD colour image, and allows the intensity estimate to be refined. It also allows them to be updated in the event that the real lighting in the scene changes. In this way, the extra cameras extend the field of view, meaning that more of the dense model's colours can be updated each frame.

First, the pose of the camera must be determined. This is inferred from the pose of the RGBD camera (obtained through the tracking approach described in section 4.5) and the relative pose of the two cameras (obtained via camera calibration). Generally, the image will not have been captured at the same time as the most recent RGBD frame, so an approximate RGBD pose at the current time is extrapolated from the two most recent RGBD poses via spherical linear interpolation.

The updating uses an index map rendered using this camera pose, in a similar way to the RGBD updating process. When searching for possible matches in the 4x4 window, the match closest to the camera is selected, providing it satisfies a number of criteria:

(1) The matched surfel lies in front of the camera, and is at least a minimum distance away (10cm).
(2) The matched surfel is not too distant from the camera (over 10m).
(3) The normal of the matched surfel points towards the camera (i.e. $c.n < 0$, where c is the camera-to-surfel vector and n is the surfel normal).
(4) The confidence of the surfel is above a threshold (5).
(5) The colour of the matched surfel is sufficiently similar to that of the new surfel.

The first and third criteria make sure that the match is actually visible to the camera. The second ensures that the surfel is not so far away that the resolution of the camera image will be insufficient to determine its colour. The final one is imposed because the location and normal of new surfels tends to vary a lot in the first few frames, meaning they may be matched incorrectly.

This updating process can also be applied when the camera used is omnidirectional, as in the case of the fisheye camera used in our prototype. In this case, an index cubemap is rendered instead of 2D index map image. The rest of the updating process is similar - for each 4x4 set of indices in the cubemap, a matching surfel is potentially found, and if so, its colour is updated using a sample from the omnidirectional image.

4.4 Environment Map Keyframes

In addition to the surfel-based component of the reconstruction, we also capture environment map keyframes. These each consist of a cubemap and 3D location, as briefly mentioned in section 3. These are intended to capture details of the real environment not present in the surfel-based dense model, either because they have not yet been observed, or they are too distant for their depths to be inferred by the RGBD camera.

The first keyframe is created at the initial location of the camera system. Its colours are initialised using the output of the cameras, and then it is inpainted using an efficient GPU-based push-pull approach [12, 29]. This inpainting is repeated each time the keyframe is updated, to ensure the inpainted regions are consistent with the rest of the keyframe. The alpha channel of the texture stores a covariance value, which is used when updating the keyframe. Pixels which have been inpainted and not yet directly observed have the alpha value set to zero.

The push-pull inpainting approach proceeds in two steps. The first, "push", step is similar to constructing a mipmap pyramid. Proceeding from fine to coarse levels, each pixel is set to be the average of all valid pixels in the next finest level (i.e. those pixels which have been observed by the fisheye camera). In the second, "pull", step, working from coarse to fine pyramid levels, each pixel which has not yet been observed is assigned the colour of its parent pixel. This process is applied to each face of the cubemap separately. In the event a face has not been observed at all, its neighbours are inpainted first, and their border pixels duplicated along the edges of the missing face before inpainting is applied.

Figure 3 shows an example of the result of the push-pull inpainting applied to an environment map keyframe, after it was updated using the first 4 bracketed fisheye images. The keyframes are shown as cube nets. In the top left image, before inpainting, the pixels which have not yet been observed or which were saturated or zero-valued are black. Note that some pixels on the window were saturated, as the fisheye auto-exposure has not yet adjusted its exposure settings. After the inpainting step, all pixels in the cubemap have HDR colour values. Note that much of the inpainted region in the lower half of the cubemap will be occupied by the dense surfel model when the final environment map is rendered (see figure 8).

Keyframes are updated whenever a new colour fisheye image is captured. Each texel of the cubemap is projected into the new image, and it is sampled and used to update the colour. When projecting, only the rotational component of the camera's pose is

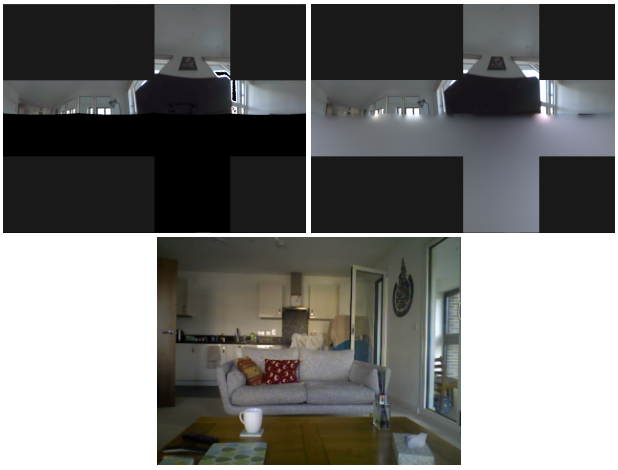

Figure 3: Environment map keyframe generated using first captured fisheye frames, before and after push-pull inpainting. Top left: keyframe before inpainting. Top right: keyframe after inpainting. Bottom: Real scene captured by RGBD camera, for comparison.

used (i.e. the points in the map are considered to be at infinity). The update is carried out using a simple Kalman filter. If the pixel to be updated is being observed for the first time (i.e. has an alpha value of zero), however, it is simply replaced with the observed colour. If the observed colour is saturated or black, no update is carried out, as the true radiance at this pixel is unknown.

The Kalman filter-based updating strikes a balance between fast updating and eliminating temporal flickering artefacts which could otherwise result from combining the bracketed exposures captured by the fisheye camera at slightly different times. The parameters of the filter can be tuned to favour either fast updating or temporal smoothness in the output. However, some artefacts can occur when rapid changes occur in the real scene - for example a light being turned off might result in a smoothed, slightly delayed change in the keyframe, and rapid moving objects can cause a ghosting effect.

A new keyframe is generated when the translational component of the camera system's pose exceeds a threshold. This helps to prevent the keyframe becoming obsolete when the distant environment assumption is violated. New keyframes are initialised with the colours from the previous keyframe, and the alpha channel set to zero (indicating the pixels have not yet been observed). Previous keyframes are retained, and if the camera moves back within range of an existing keyframe, that keyframe will subsequently be updated and used in rendering.

When new surfels are added, the environment keyframe can be consulted to obtain HDR colours for them, where available. We do so when a newly added surfel is either saturated or below the black point in the RGBD image, meaning its intensity would otherwise be unknown. The surfel is projected back into the environment map keyframe to find a colour, and this colour is used if the alpha channel is non-zero (i.e. this part of the keyframe has been observed by a camera, and was not inpainted or propagated from a previous keyframe).

4.5 Camera Tracking

Similar to other dense RGBD SLAM approaches, the location of the RGBD camera is tracked via an iterative dense frame-to-model alignment process as each new frame is observed. Our camera tracking implementation makes use of the colour and depth tracking approach of Kerl et al. [25], adapted to make use of the HDR information stored in the surfels.

Specifically, we use the RGBD alignment process developed by Kerl et al. [25, 26, 46]. Given two pairs of intensity and depth images, this finds the rigid transform which best aligns them, in the sense that it minimises the colour and intensity differences.

We use a similar overall frame-to-model tracking approach to earlier dense RGBD reconstruction methods including Kinect Fusion [33]. When each new pair of colour and depth images is captured by the RGBD camera, we render the current surfel model from the previous pose, and align the new depth and colour frames to these rendered frames. This provides an estimated transform for the current frame, which can be composed with the previous camera pose estimate to provide an updated camera pose.

In contrast to earlier approaches, we make use of the HDR information available in our model. When rendering the colour frame, we apply the current estimated exposure and white balance settings of the RGBD colour camera, determined as outlined in section 4.2. This ensures that the effective exposure and white balance in the real and rendered colour images are the same. The colour images are then converted to grayscale intensity values, and the intensity and depth images aligned using the approach of Kerl et al. to provide the final transform.

Tracking using HDR colours and depths brings a number of advantages over earlier approaches such as Kinect Fusion [33], which tracks using depth alone. Should the depth image lack sufficient geometric detail to enable the tracking to function (e.g. the camera observes a flat wall), the colour tracker is still often able to successfully produce an estimate. Conversely, if colour tracking fails (for example, due to a change in lighting conditions), the depth tracking is often able to successfully recover a camera transform. Since we track using HDR colours, rather than the LDR colours used by InfiniTAM [18], for example, it is also possible to enable the auto exposure and white balance features of the colour camera without adversely affecting the colour tracking. This allows the camera to capture a greater number of correctly exposed pixels, and to capture a more pleasing image to present to the end user.

4.6 Dynamic Geometry

Many previous RGBD reconstruction approaches, including that of Keller et al. [24] have focused on reconstructing an accurate model of a static real environment, and contain measures to avoid adding dynamic real objects to the model. However, the intention of our approach is to respond in real time to changes in geometry, enabling virtual content in MR applications to (for example) reflect dynamic real geometry, such as the user's hand. We achieve this by building upon the approach of Keller et al. in a number of ways.

Firstly, we separate the criteria when using surfels for tracking and MR. Only those surfels that qualify as stable (based on the confidence threshold detailed in section 4.2) are used for tracking. However, all surfels are used when rendering for MR (for example,

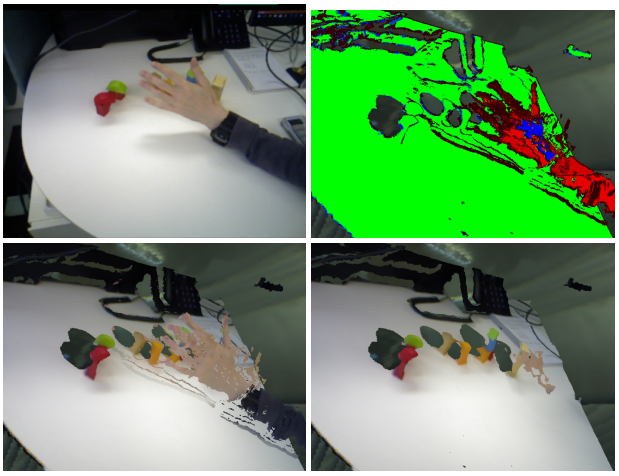

Figure 4: Examples of surfels used for tracking vs. surfels used for rendering MR content

when rendering environment maps used to simulate reflection from a virtual object). This enables the MR rendering to respond instantly to geometric changes, without impacting the quality of the camera tracking.

Figure 4 shows an example of the difference between these categories. The top image is the input colour image from the RGBD camera. The bottom left image shows a rendering of all the surfels captured by the approach. The bottom right image shows just those surfels which are categorised as stable. The top right image shows a visualisation of the confidence values of the surfels: here shades of red are unstable, and stable surfels range in colour from blue to green as confidence increases. Here, the hand has just moved into the frame. It is present in the surfel model, and so will correctly impact the lighting of virtual objects, but is not present in the set used for tracking, so will not negatively impact the tracking.

Surfel colours are generally updated using a rolling-average approach, similar to that used by other approaches such as [33] and [24]. This approach is helpful in eliminating noise present in the colour images by averaging the colours of multiple observations of each surfel. However, since we allow environment lighting to change, the surfel colours need to be updated rapidly if, for example, a light is turned on or a window opened. To account for this, if a newly observed colour differs significantly from the existing surfel, its colour is replaced with the new value and its confidence is reduced.

In order to respond quickly to dynamic objects, it is also critical to quickly remove surfels which no longer correspond to valid real geometry. This allows the model to better reflect reality in the event that a real object is moved or removed. Keller et al. [24] implement one method for achieving this, which we also employ. This involves searching for those surfels which have a smaller depth than that observed in the current frame. Since we assume surfels are opaque, this is a physically impossible situation, and indicates that these surfels should be deleted from the model. These surfels are identified and removed during the updating stage, using the index map.

Figure 5: Surfel reconstructions after sliding a real piece of coloured paper across a table.

Figure 6: Surfel reconstructions before and after removing a box from the real scene.

In some situations, this method fails to remove bad surfels. This is a particular problem when depth values are missing, or when depth is insufficient to identify the problem (e.g. a piece of paper is moved along a tabletop). Since depth values frequently cannot be obtained, small clusters of erroneous surfels are often left behind in the model.

We address this problem by also checking for colour violations. These are points where the colour of the visible surface of the model differs from that actually observed by the camera. These are identified by rendering a view of the model from the current camera pose, and comparing the HDR colours with the real image at each pixel. At pixels where the colour is sufficiently different, a colour violation is detected and the corresponding surfel is marked for deletion.

In order to improve efficiency and avoid removing surfels erroneously, the colour violation check is not carried out at pixel locations where a new surfel was added, or an existing surfel was successfully updated.

Figures 5 and 6 show examples of how each of these approaches improves the updating of the surfel model. Figure 5 shows the surfel reconstruction after a green piece of paper has been moved from left to right over a tabletop. In the left example, standard colour updating was used, and the colours are slowly updating, leaving an afterimage of the paper's previous position. In the right example, our faster colour updating approach was used and the colours were updated instantly, removing this artefact.

Figure 6 shows the importance of removing depth and colour violations from the surfel model. The top left image shows the initial surfel reconstruction of a box placed on a table. The other images show the state of the reconstruction a few frames after the box has been removed. In the top right example, neither colour nor depth violation removal was used, and the surfels corresponding to the box remain in the model. In the bottom left example, just depth violations were removed. Much of the box is gone from the model, but the lower half had depths similar to those of the table below, so these could not be classified as depth violations and removed. In the bottom right, removing both depth and colour violations has successfully removed nearly all of the outdated surfels corresponding to the box from the model.

4.7 Rendering Virtual Content

In order to demonstrate how the captured models could be used to illuminate virtual content in MR, a video see-through application was developed. This made use of both models to illuminate a simple virtual scene, and add this to the colour images produced by the RGBD camera.

The application first rendered a HDR environment map from the location of the virtual objects. This was achieved by first copying the current environment map keyframe, and then rendering the dense surfel model on top of this. When rendering the surfel model, backface culling was disabled, as it is common for only the side of an object facing away from the virtual content to be observed by the camera system.

The virtual content can then be rendered, using this environment map. The current camera position as determined via the RGBD tracking approach is used, with camera parameters set using the real camera calibration. In our implementation, the rendering is achieved using simple environment mapping [10] for reflective objects and diffuse differential precomputed radiance transfer (PRT) [45] for diffuse objects and cast shadows. During precomputation, we assume the virtual object will be placed on a Lambertian planar surface. Note, however, that the lighting information captured by our approach could be used as input to many other rendering approaches, such as those employed in [38].

The virtual content is rendered into a floating point HDR texture, which is then appropriately combined with the HDR version of the input RGBD image. In order to produce the final image to display to the user, the auto exposure, white balance and gamma of the RGBD camera are then applied, mapping this back to a LDR output. In this way, the output appears as if it were observed by the real RGBD camera.

5 RESULTS

The application described in section 4 was tested in a variety of real scenarios, with a number of different virtual objects. Figure 7 shows examples of a number of different virtual objects, demonstrating some of the virtual materials which can be effectively simulated using the approach.

Figure 8 shows an example where the system was used in a scene with a wide dynamic range, demonstrating the ability of our approach to function correctly in such scenes. This scene was captured in a flat on a sunny day, and the windows are significantly brighter than the rest of the scene, however the bracketing approach outlined in section 4.1 enables the full dynamic range of the scene to be captured. Additionally, as the camera is moved around the real table, its auto exposure adjusts due to the bright windows. This change in auto exposure is detected, and the brightness of the virtual bunny adjusted appropriately when mapping the output to LDR. This means the bunny appears darker, and remains consistent with the real scene around it.

Figure 7: Examples of MR frames rendered using the approach, containing a variety of different virtual objects. Top: virtual pan, with ideal (mirror) reflection. Bottom left: virtual terracotta bunny, with diffuse surface. Bottom right: virtual trumpet, with sligthly rough, coloured reflective surface.

Note that the results shown here were all captured in indoor environments. This was mainly due to the RGBD camera used in our system, which was unable to produce accurate depth measurements in direct sunlight, and not due to a limitation of the approach itself. Depth cameras are now becoming more capable of functioning in outdoor environments, and should allow our approach to be used in a wider variety of settings in the future.

5.1 Comparison with Real Object

Figure 9 shows an example comparing a real chrome sphere with a virtual sphere rendered using our approach. The image with the real chrome sphere was first captured with the camera pair attached to a tripod via a quick release mount. The chrome sphere was then removed, and the RGBD camera was detached from the tripod and moved around the scene briefly to reconstruct the real environment near the sphere. The camera was then replaced on the tripod, and our approach was then used to render a virtual sphere of the same size in the same location as the real sphere, enabling both to be directly compared.

The virtual and real spheres appear similar overall, with nearby reflected objects located correctly in the virtual reflective sphere, and of similar appearance. There is some difference in the colour of the two spheres - in particular, the real sphere is somewhat darker. This is due mainly to the fact that the virtual sphere was rendered

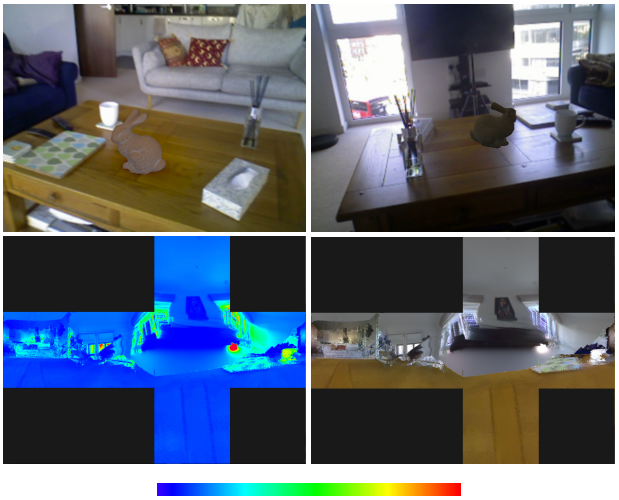

Figure 8: Example using the system in an environment with a wide dynamic range. Top left: view of virtual bunny. Top right: view of virtual bunny from other side of table. Note that the auto exposure has adjusted to compensate for the bright windows, but the bunny remains consistent with the real scene. Bottom left: HDR radiance environment map used to render bunny, shown as heatmap (with colour key below) Bottom right: Environment map in colour, with RGBD exposure settings from top left image applied. Note that the windows are much brighter than the rest of the scene.

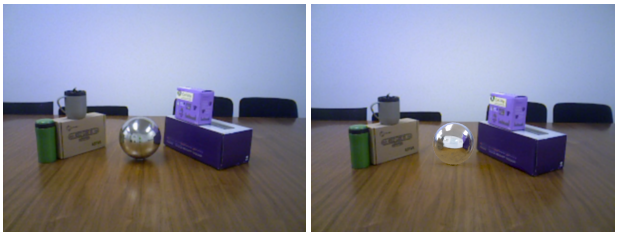

Figure 9: Comparison of a virtual reflective sphere rendered using the application with a real chrome sphere of the same size.

with an ideal mirror surface, wheras the real sphere is an imperfect reflector. Additionally, the real sphere has a sharper shadow than the virtual sphere. This is due to the shadow using PRT, with only 4 SH bands, and consequently being unable to capture the higher frequencies of the incident lighting.

There are slight differences in the positions of nearby real objects due to the use of environment mapping. The reflections on the sphere were simulated using an environment map rendered from the centre of the sphere, which results in slight geometric inaccuracies when this map is used to simulate reflections on the surface of the sphere.

The most noticeable difference between the two spheres is in the positioning of the distant geometry. This is slightly different

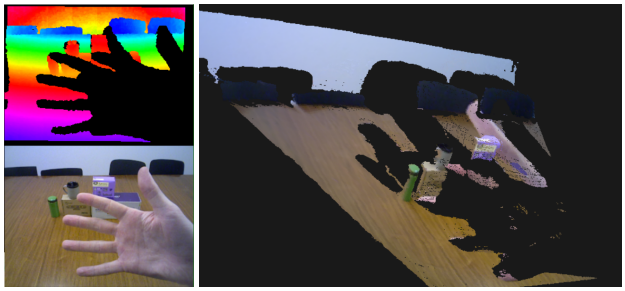

Figure 10: Examples of a failure case in the surfel reconstruction. Surfels have been incorrectly removed from the model when a user placed their hand too close to the RGBD camera. Left: depth and colour inputs. Right: rendering of surfel reconstruction.

due to being rendered using the environment map keyframe, which was captured at a different point in space (i.e. at the fisheye camera location). This slight geometric inaccuracy could be removed using an approach such as [47], however their approach requires the distant geometry to be known a priori. Our approach accepts this slight inaccuracy in order to produce perceptually plausible results in unprepared environments.

5.2 Limitations

The current approach produces plausible results in a number of settings, but does suffer from some limitations. Since the system relies upon the dense surfel reconstruction and environment map keyframes, it may produce inaccurate results when these are incomplete. This can be problematic when the depth camera cannot infer depths for some objects in the real scene, for example if a real object is translucent or highly specular. Such objects will not be present in the reconstruction, and will therefore not be present in reflections from virtual objects added to the scene. This problem is common to most dense reconstruction approaches using RGBD cameras, and is not easily addressed with current depth sensing technology.

The colour violation removal approach detailed in section 4.6 can sometimes erroneously remove valid surfels from the reconstruction. Small numbers of surfels can be removed due to slight tracking inaccuracies or non-lambertian real objects (e.g. specular highlights). The problem is most noticeable, however, when a real object moves very close to the RGBD camera. Figure 10 shows an example of this issue, where a user's hand has moved very close to the RGBD camera. Here, the camera was unable to find depth values for the hand since it fell outside the depth range of the camera. As a result, all the surfels behind the hand are labelled as colour violations and removed. This problem would be partly mitigated by using a more recent RGBD camera with a lower minimum depth.

6 CONCLUSION

We have presented a new approach for capturing real illumination information for mixed reality applications using a compact camera system, and demonstrated how the information can be used to

render realistic virtual content. The advantages of the approach include its ability to provide results immediately, without prior scene knowledge, and to update the lighting quickly to reflect changes in the real environment. In addition to lighting information, the approach also captures a dense model of the real scene useful for a range of MR applications.

We feel that approaches such as this, that are capable of constructing and updating dense models with real geometric and illumination data will be an important component enabling future MR applications to add content which seamlessly blends in with the real world.

In the future, it would be interesting to explore how recent machine learning techniques could be used to improve the results of our approach. For example, a method for completing missing regions of dense RGBD reconstructions such as [7] could be applied to provide a more complete dense model more quickly. One could also explore using the output of our approach to determine semantic information about the environment; for example, if the approach is being used in an indoor or outdoor setting. This information could then be used to tailor the added virtual content to the real setting.

ACKNOWLEDGMENTS

This research was funded by the UK's EPSRC via a UCL EngD VEIV studentship (grant reference EP/G037159/1), and by Imagination Technologies Limited. The authors would also like to thank Tobias Ritschel for his helpful suggestions on improving the RGBD SLAM method.

REFERENCES

[1] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. 2003. Photorealistic rendering for augmented reality using environment illumination. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*. IEEE, 208–216.

[2] Miika Aittala. 2010. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer* 26, 6-8 (2010), 669–678.

[3] Ibrahim Arief, Simon McCallum, and Jon Yngve Hardeberg. 2012. Realtime estimation of illumination direction for augmented reality on mobile devices. In *Color and Imaging Conference*, Vol. 2012. Society for Imaging Science and Technology, 111–116.

[4] Dan A Calian, Kenny Mitchell, Derek Nowrouzezahrai, and Jan Kautz. 2013. The shading probe: fast appearance acquisition for mobile AR. In *SIGGRAPH Asia 2013 Technical Briefs*. ACM, 20.

[5] D. Caruso, J. Engel, and D. Cremers. 2015. Large-Scale Direct SLAM for Omnidirectional Cameras. In *International Conference on Intelligent Robots and Systems (IROS)*.

[6] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 24.

[7] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. 2018. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

[8] Paul Debevec. 1998. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 189–198.

[9] Simon Gibson and Alan Murta. 2000. Interactive Rendering with Real-World Illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*. Springer-Verlag, 365–376.

[10] Ned Greene. 1986. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications* 6, 11 (1986), 21–29.

[11] Thorsten Grosch, Tobias Eble, and Stefan Mueller. 2007. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. ACM, 125–132.

[12] Jeffrey P Grossman and William J Dally. 1998. Point sample rendering. In *Rendering techniques*. Springer, 181–192.

[13] Lukas Gruber, Tobias Langlotz, Pradeep Sen, Tobias Höherer, and Dieter Schmalstieg. 2014. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality (VR)*. IEEE, 15–20.

[14] Lukas Gruber, Thomas Richter-Trummer, and Dieter Schmalstieg. 2012. Real-time photometric registration from arbitrary geometry. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*. IEEE, 119–128.

[15] Lukas Gruber, Jonathan Ventura, and Dieter Schmalstieg. 2015. Image-space illumination for augmented reality in dynamic environments. In *2015 IEEE Virtual Reality (VR)*. IEEE, 127–134.

[16] Michael Haller, Stephan Drab, and Werner Hartmann. 2003. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 56–65.

[17] Gentaro Hirota, David T Chen, William F Garrett, Mark A Livingston, et al. 1996. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 429–438.

[18] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S Torr, and D. W. Murray. 2015. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. *IEEE Transactions on Visualization and Computer Graphics* 22, 11 (2015).

[19] Peter Kán and Hannes Kaufmann. 2012. Physically-Based Depth of Field in Augmented Reality.. In *Eurographics (Short Papers)*. 89–92.

[20] Paul Kan and Hannes Kaufmann. 2013. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. IEEE, 133–141.

[21] Masayuki Kanbara and Naokazu Yokoya. 2004. Real-time Estimation of Light Source Environment for Photorealistic Augmented Reality.. In *ICPR (2)*. Citeseer, 911–914.

[22] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. 2011. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 157.

[23] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. 2014. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics (TOG)* 33, 3 (2014), 32.

[24] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. 2013. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision-3DV 2013, 2013 International Conference on*. IEEE, 1–8.

[25] Christian Kerl, Jurgen Sturm, and Daniel Cremers. 2013. Dense visual SLAM for RGB-D cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. Citeseer, 2100–2106.

[26] Christian Kerl, Jürgen Sturm, and Daniel Cremers. 2013. Robust odometry estimation for RGB-D cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 3748–3754.

[27] Kenneth Levenberg. 1944. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics* 2, 2 (1944), 164–168.

[28] Donald W Marquardt. 1963. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics* 11, 2 (1963), 431–441.

[29] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. 2007. Efficient Point-Based Rendering Using Image Reconstruction.. In *SPBG*. 101–108.

[30] MATLAB. 2017. *MATLAB R2017a: Computer Vision Toolbox*. The MathWorks Inc., Natick, Massachusetts.

[31] Maxime Meilland, Christian Barat, and Andrew Comport. 2013. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. IEEE, 143–152.

[32] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. 2018. LIME: Live Intrinsic Material Estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 11. http://gvv.mpi-inf.mpg.de/projects/LIME/

[33] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.

[34] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)* 32, 6 (2013), 169.

[35] Toshiya Okabe, Imari Sato, and Yoichi Sato. 2004. Spherical harmonics vs. haar wavelets: Basis for recovering illumination from cast shadows. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, Vol. 1. IEEE, I–50.

[36] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual*

conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 335–342.

[37] Kai Rohmer, Wolfgang Buschel, Raimund Dachselt, and Thorsten Grosch. 2015. Interactive Near-Field Illumination for Photorealistic Augmented Reality with Varying Materials on Mobile Devices. *Visualization and Computer Graphics, IEEE Transactions on* 21, 12 (2015), 1349–1362.

[38] Kai Rohmer, Johannes Jendersie, and Thorsten Grosch. 2017. Natural Environment Illumination: Coherent Interactive Augmented Reality for Mobile and non-Mobile Devices. *IEEE transactions on visualization and computer graphics* 23, 11 (2017), 2474–2484.

[39] Henry Roth and Marsette Vona. 2012. Moving Volume KinectFusion.. In *BMVC*, Vol. 20. 1–11.

[40] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. 2014. Dense planar SLAM. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE, 157–164.

[41] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. 1999. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *Visualization and Computer Graphics, IEEE Transactions on* 5, 1 (1999), 1–12.

[42] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. 2001. Stability issues in recovering illumination distribution from brightness in shadows. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 2. IEEE, II–400.

[43] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. 2003. Illumination from shadows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25, 3 (2003), 290–300.

[44] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. 2006. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5695–5701.

[45] Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, Vol. 21. ACM, 527–536.

[46] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. 2011. Real-time visual odometry from dense RGB-D images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 719–722.

[47] David R Walton, Diego Thomas, Anthony Steed, and Akihiro Sugimoto. 2017. Synthesis of environment maps for mixed reality. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 72–81.

[48] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. 2015. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research* 34, 4-5 (2015), 598–626.

[49] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. 2015. ElasticFusion: Dense SLAM without a pose graph. Robotics: Science and Systems.

[50] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. 2016. ElasticFusion: Real-Time Dense SLAM and Light Source Estimation. *Intl. J. of Robotics Research, IJRR* (2016).

[51] Yiying Yao, Hidenori Kawamura, and Akira Kojima. 2012. Shading derivation from an unspecified object for augmented reality. In *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 57–60.

[52] Xianghua Ying and Zhanyi Hu. 2004. Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model. In *European Conference on Computer Vision*. Springer, 442–455.

[53] Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. 2013. Octree-based fusion for realtime 3D reconstruction. *Graphical Models* 75, 3 (2013), 126–136.

[54] Edward Zhang, Michael F Cohen, and Brian Curless. 2016. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 174.

[55] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J Mitra. 2012. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph.* 31, 4 (2012), 99–1.